

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE TOD

OBJECT MODULE PLACED IN TOD.OBJ

COMPILER INVOKED BY: :FO: TOD.PLN DATE(APRIL 1, 1982) XREF OPTIMIZE(3) DEBUG

```

$compact
$title ('CP/M-86 1.1 Time and Date')
1  tod:
  do;
/*
    revised by DH 9/17/81
    color removed by DH 10/06/81
    display gets ascii date (not rolander) - DH 10/27/81 */

/* compiled with following submit under VAX VMS

    set verify
    set def [doug.cpm86]
    $ plm86 'p1'.plm 'p2' 'p3' 'p4' optimize(3) debug
    $ link86 mcd.obj,'p1'.obj to 'p1'.lnk
    $ loc86 'p1'.lnk od(sm(dats,code,data,stack,const)) ad(sm(dats(0),code(0))) ss(stack(+16))
    $ h86 'p1'

*/

2  1  declare dcl literally 'declare';
3  1  dcl lit literally 'literally';
4  1  dcl forever lit 'while 1';

5  1  dcl CPMproduct lit '00h';
6  1  dcl cpmversion lit '22h';
7  1  dcl CR lit '0dh',
      LF lit '0ah';

8  1  mon1:
      procedure (func,info) external;
9  2      declare func byte;
10 2      declare info address;
11 2      end mon1;

12 1  mon2:
      procedure (func,info) byte external;
13 2      declare func byte;
14 2      declare info address;
15 2      end mon2;

16 1  mon3:
      procedure (func,info) address external;
17 2      declare func byte;
18 2      declare info address;
19 2      end mon3;

20 1  mon4:
      procedure (func,info) pointer external;

```

CP/M-86 v.1.1 (Vol. IV)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # C81-162

```
21 2      declare func byte;
22 2      declare info address;
23 2      end mon4;

24 1      declare fcb (1) byte external;
25 1      declare fcb16 (1) byte external;
26 1      declare buff (1) byte external;

27 1      read$console:
          procedure byte;
28 2          return mon2 (1,0);
29 2      end read$console;

30 1      write$console:
          procedure (char);
31 2          declare char byte;
32 2          call mon1 (2,char);
33 2      end write$console;

34 1      print$buffer:
          procedure (buffer$address);
35 2          declare buffer$address address;
36 2          call mon1 (9,buffer$address);
37 2      end print$buffer;

38 1      check$console$status:
          procedure byte;
39 2          return mon2 (11,0);
40 2      end check$console$status;

41 1      version:
          procedure address;
42 2          return mon3(12,0);
43 2      end version;

44 1      terminate:
          procedure;
45 2          call mon1 (0,0);
46 2      end terminate;

47 1      get$sysdat:
          procedure pointer;
48 2          return (mon4(49,0));
49 2      end get$sysdat;

50 1      crlf:
          procedure;
51 2          call write$console (0dh);
52 2          call write$console (0ah);
53 2      end crlf;

54 1      put$blanks:
          procedure;
55 2          declare i byte;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # _____

```

56 2      i=3;
57 2      do while (i:=i-1) < 0;
58 3      call write$console (' ');
59 3      end;
60 2      end put$blanks;

61 1      ENTRY$ERROR:
        procedure;

62 2      CALL PRINT$BUFFER(.(CR,LF,'Invalid Date & Time Format','$'));
63 2      CALL PRINT$BUFFER(.(CR,LF,CR,LF,
        'Please retry using:',CR,LF,CR,LF,
        ' TOD MM/DD/YY HH:MM:SS','$'));
64 2      CALL TERMINATE;
65 2      end ENTRY$ERROR;

```

/*****

Time & Date ASCII Conversion Code

*****/

```

66 1      declare tod$adr address;
67 1      declare tod based tod$adr structure (
        opcode byte,
        date address,
        hrs byte,
        min byte,
        sec byte,
        ASCII (21) byte );

68 1      declare string$adr address;
69 1      declare string based string$adr (1) byte;
70 1      declare index byte;

```

```

71 1      emitchar: procedure(c);
72 2      declare c byte;
73 2      string(index := index + 1) = c;
74 2      end emitchar;

```

```

75 1      emitn: procedure(a);
76 2      declare a address;
77 2      declare c based a byte;
78 2      do while c < '$';
79 3      string(index := index + 1) = c;
80 3      a = a + 1;
81 3      end;
82 2      end emitn;

```

```

83 1      emit$bcd: procedure(b);
84 2      declare b byte;
85 2      call emitchar('0'+b);
86 2      end emit$bcd;

```

```

87 1      emit$bcd$pair: procedure(b);

```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

88 2      declare b byte;
89 2      call emit$bcd(shr(b,4));
90 2      call emit$bcd(b and 0fh);
91 2      end emit$bcd$pair;

92 1      emit$colon: procedure(b);
93 2      declare b byte;
94 2      call emit$bcd$pair(b);
95 2      call emitchar(':');
96 2      end emit$colon;

97 1      emit$bin$pair: procedure(b);
98 2      declare b byte;
99 2      call emit$bcd(b/10);
100 2     call emit$bcd(b mod 10);
101 2     end emit$bin$pair;

102 1     emit$slant: procedure(b);
103 2     declare b byte;
104 2     call emit$bin$pair(b);
105 2     call emitchar('/');
106 2     end emit$slant;

107 1     declare chr byte;

108 1     gnc: procedure;
        /* get next command byte */
109 2     if chr = 0 then return;
111 2     if index = 20 then
112 2     do;
113 3         chr = 0;
114 3         return;
115 3     end;
116 2     chr = string(index := index + 1);
117 2     end gnc;

118 1     deblank: procedure;
119 2         do while chr = ' ';
120 3         call gnc;
121 3         end;
122 2     end deblank;

123 1     numeric: procedure byte;
        /* test for numeric */
124 2     return (chr - '0') < 10;
125 2     end numeric;

126 1     scan$numeric: procedure(lb,ub) byte;
127 2     declare (lb,ub) byte;
128 2     declare b byte;
129 2     b = 0;
130 2     call deblank;
131 2     if not numeric then call entry$error;
133 2     do while numeric;
134 3     if (b and 111040000b) <> 0 then call entry$error;
136 3     b = shl(b,3) + shl(b,1); /* b = b * 10 */
137 3     if carry then call entry$error;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. B X 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

139 3      b = b + (chr - '0');
140 3      if carry then call entry$error;
142 3      call gnc;
143 3      end;
144 2      if (b < lb) or (b > ub) then call entry$error;
146 2      return b;
147 2      end scan$numeric;

148 1      scan$delimiter: procedure(d,lb,ub) byte;
149 2      declare (d,lb,ub) byte;
150 2      call deblank;
151 2      if chr <> d then call entry$error;
153 2      call gnc;
154 2      return scan$numeric(lb,ub);
155 2      end scan$delimiter;

156 1      declare
      base$year lit '78', /* base year for computations */
      base$day lit '0', /* starting day for base$year 0..6 */
      month$size (*) byte data
      /* jan feb mar apr may jun jul aug sep oct nov dec */
      ( 31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31),
      month$days (*) word data
      /* jan feb mar apr may jun jul aug sep oct nov dec */
      ( 000,031,059,090,120,151,181,212,243,273,304,334);

157 1      leap$days: procedure(y,m) byte;
158 2      declare (y,m) byte;
      /* compute days accumulated by leap years */
159 2      declare yp byte;
160 2      yp = shr(y,2); /* yp = y/4 */
161 2      if (y and 11b) = 0 and month$days(m) < 59 then
      /* y not 00, y mod 4 = 0, before march, so not leap yr */
162 2      return yp - 1;
      /* otherwise, yp is the number of accumulated leap days */
163 2      return yp;
164 2      end leap$days;

165 1      declare word$value word;

166 1      get$next$digit: procedure byte;
      /* get next lsd from word$value */
167 2      declare lsd byte;
168 2      lsd = word$value mod 10;
169 2      word$value = word$value / 10;
170 2      return lsd;
171 2      end get$next$digit;

172 1      bcd:
      procedure (val) byte;
173 2      declare val byte;
174 2      return shl((val/10),4) + val mod 10;
175 2      end bcd;

176 1      declare (month, day, year, hrs, min, sec) byte;

177 1      set$date$time: procedure;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # _____


```

178 2      declare
          (i, leap$flag) byte; /* temporaries */
179 2      month = scan$numeric(1,12) - 1;
          /* may be feb 29 */
180 2      if (leap$flag != month = 1) then i = 29;
182 2          else i = month$size(month);
183 2      day = scan$delimiter('/',1,i);
184 2      year = scan$delimiter('/',base$year,99);
          /* ensure that feb 29 is in a leap year */
185 2      if leap$flag and day = 29 and (year and 11b) <> 0 then
186 2          /* feb 29 of non-leap year */ call entry$error;
          /* compute total days */
187 2      tod.date = month$days(month)
          + 365 * (year - base$year)
          + day
          - leap$days(base$year,0)
          + leap$days(year,month)

188 2      tod.hrs = bcd (scan$numeric(0,23));
189 2      tod.min = bcd (scan$delimiter(':',0,59));
190 2      if tod.opcode = 2 then
          /* date, hours and minutes only */
191 2      do;
192 3          if chr = ':'
          then i = scan$delimiter(':',0,59);
194 3          tod.sec = 0;
195 3      end;
          /* include seconds */
196 2      else tod.sec = bcd (scan$delimiter(':',0,59));

197 2      end set$date$time;

198 1      bcd$pair: procedure(a,b) byte;
199 2      declare (a,b) byte;
200 2      return shl(a,4) or b;
201 2      end bcd$pair;

202 1      compute$year: procedure;
          /* compute year from number of days in word$value */
203 2      declare year$length word;
204 2      year = base$year;
205 2      do forever;
206 3          year$length = 365;
207 3          if (year and 11b) = 0 then /* leap year */
208 3              year$length = 366;
209 3          if word$value <= year$length then
210 3              return;
211 3          word$value = word$value - year$length;
212 3          year = year + 1;
213 3      end;
214 2      end compute$year;

215 1      declare
          week$day byte, /* day of week 0 ... 6 */
          day$list (*) byte data
          ('Sun$Mon$Tue$Wed$Thu$Fri$Sat$'),
          leap$bias byte; /* bias for feb 29 */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

216 1  compute$month: procedure;
217 2      month = 12;
218 2      do while month > 0;
219 3          if (month := month - 1) < 2 then /* jan or feb */
220 3              leap$bias = 0;
221 3              if month$days(month) + leap$bias < word$value then return;
223 3              end;
224 2      end compute$month;

225 1  declare
      date$test byte, /* true if testing date */
      test$value word; /* sequential date value under test */

226 1  get$date$time: procedure;
      /* get date and time */
227 2      hrs = tod.hrs;
228 2      min = tod.min;
229 2      sec = tod.sec;
230 2      word$value = tod.date;
      /* word$value contains total number of days */
231 2      week$day = (word$value + base$day - 1) mod 7;
232 2      call compute$year;
      /* year has been set, word$value is remainder */
233 2      leap$bias = 0;
234 2      if (year and 11b) = 0 and word$value > 59 then
235 2          /* after feb 29 on leap year */ leap$bias = 1;
236 2      call compute$month;
237 2      day = word$value - (month$days(month) + leap$bias);
238 2      month = month + 1;
239 2      end get$date$time;

240 1  emit$date$time: procedure;
241 2      call emitn(.day$list(shl(week$day,2)));
242 2      call emitchar(' ');
243 2      call emit$slant(month);
244 2      call emit$slant(day);
245 2      call emit$bin$pair(year);
246 2      call emitchar(' ');
247 2      call emit$colon(hrs);
248 2      call emit$colon(min);
249 2      call emit$bcd$pair(sec);
250 2      end emit$date$time;

251 1  tod$ASCII:
      procedure (parameter);
252 2      declare parameter address;
253 2      declare ret address;

254 2      ret = 0;
255 2      tod$adr = parameter;
256 2      string$adr = .tod.ASCII;
257 2      if tod.opcode = 0 then
258 2          do;
259 3              call get$date$time;
260 3              index = -1;
261 3              call emit$date$time;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. B. X 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

262 3      end;
        else
263 2      do;
264 3          if (tod.opcode = 1) or
              (tod.opcode = 2) then
265 3              do;
266 4                  chr = string(index:=0);
267 4                  call set$date$time;
268 4                  ret = ,string(index);
269 4              end;
        else
270 3          do;
271 4              call entry$error;
272 4          end;
273 3      end;
274 2      end tod$ASCII;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. Box 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

/*****
*****/

```

```

275 1      declare tod$pointer pointer;
276 1      declare tod$ptr structure (
              offset word,
              segment word) at (@tod$pointer);
277 1      declare extrnl$tod based tod$pointer structure (
              date address,
              hrs byte,          /* in system data area */
              min byte,
              sec byte,
              ASCII (17) BYTE );

278 1      declare lcltod structure (          /* local to this program */
              opcode byte,
              date address,
              hrs byte,
              min byte,
              sec byte,
              ASCII (21) byte );

279 1      DECLARE TOD$DELIMITER LIT 'LCLTOD.ASCII(8)';

280 1      declare (i,c) byte;
281 1      declare ret address;

282 1      display$tod:
              procedure;

283 2          lcltod.opcode = 0;          /* read tod */
/*          call movb (@extrnl$tod.date,@lcltod.date,5);
          call tod$ASCII (.lcltod); */
284 2          call write$console (CR);
/*          call write$console (1bh);
          call write$console ('b');
          call write$console (3);      */
285 2          do i = 0 to 16;
286 3              if (c:=extrnl$tod.ascii(i)) <> ',' then
287 3                  call write$console (c);

```

```

                else
288 3          call put$blanks;
289 3          end;
290 2      end display$tod;

```

```

/*
Main Program
*/

```

```

291 1      declare tod$sd$offset lit '1BH'; /* offset of TOD structure in data page */
292 1      declare vers address;
293 1      declare last$dseg$byte byte
          initial (0);

294 1      plm:
          procedure public;
295 2          vers = version;
296 2          if low (vers) < cpm$version or high (vers) < cpm$product then
297 2              do;
298 3                  call print$buffer (.(CR,LF,'Requires CP/M-86 2.2','$'));
299 3                  call terminate;
300 3              end;

301 2          tod$pointer = get$sysdat;
302 2          tod$ptr.offset = tod$ptr.offset + tod$sd$offset;
303 2          if (fcb(1) < ' ') and (fcb(1) < 'P') then
304 2              do;
305 3                  i=0;
306 3                  do while (buff(i:=i+1) = ' ');
307 4                      end;
308 3                  call move (21,.buff(i),.lcltod.ASCII);
309 3                  lcltod.opcode = 1;
310 3                  IF TOD$DELIMITER < ' ' AND TOD$DELIMITER < ',' THEN
311 3                      CALL ENTRY$ERROR;
312 3                  if lcltod.ascii(2) < '/' or lcltod.ascii(5) < '/' or
                      lcltod.ascii(11) < ':' or lcltod.ascii(14) < ':' then
313 3                      call entry$error;
314 3                  call tod$ASCII (.lcltod);
315 3                  call print$buffer (.(
                      'Strike key to set time','$'));
316 3                  ret = read$console;
317 3                  call movb (@lcltod.date,@extrnl$tod.date,5); /* use pl/m-86 move */
318 3                  TOD$DELIMITER = ',';
319 3                  call movb (@lcltod.ASCII,@extrnl$tod.ASCII,17); /* move into data pg */
320 3                  call crlf;
321 3              end;
322 2          do while fcb(1) = 'P';
323 3              call display$tod;
324 3              if check$console$status then
325 3                  do;
326 4                  ret = read$console;
327 4                  fcb(1) = 0;
328 4              end;
329 3          end;
330 2          call display$tod;
/*      call write$console (lbh);

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. ... X 579

PACIFIC ... A 93950

SER. #

```
call write#console ('b');  
call write#console (7); #/  
331 2    call terminate;  
332 2    end plm;  
  
333 1    end tod;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # _____

CROSS-REFERENCE LISTING

 DEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

75	0004H	2	A.	WORD PARAMETER AUTOMATIC	76	77	78	79	80
198	0006H	1	A.	BYTE PARAMETER AUTOMATIC	199	200			
272	0006H	21	ASCII.	BYTE ARRAY(21) MEMBER(LCLTOD)	308	310	312	318	319
67	0006H	21	ASCII.	BYTE ARRAY(21) MEMBER(TOD)	256				
277	0005H	17	ASCII.	BYTE ARRAY(17) MEMBER(EXTNLTO)		286	319		
128	0017H	1	B.	BYTE	129	134	136	139	144
87	0004H	1	B.	BYTE PARAMETER AUTOMATIC	88	89	90		
92	0004H	1	B.	BYTE PARAMETER AUTOMATIC	93	94			
97	0004H	1	B.	BYTE PARAMETER AUTOMATIC	98	99	100		
198	0004H	1	B.	BYTE PARAMETER AUTOMATIC	199	200			
102	0004H	1	B.	BYTE PARAMETER AUTOMATIC	103	104			
83	0004H	1	B.	BYTE PARAMETER AUTOMATIC	84	85			
156			BASEDAY.	LITERALLY	231				
156			BASEYEAR.	LITERALLY	184	187	204		
172	02A1H	39	BCD.	PROCEDURE BYTE STACK=0006H		188	189	196	
198	03E9H	17	BCDPAIR.	PROCEDURE BYTE STACK=0006H					
26	0000H	1	BUFF.	BYTE ARRAY(1) EXTERNAL(6)	306	308			
34	0004H	2	BUFFERADDRESS. . .	WORD PARAMETER AUTOMATIC	35	36			
77	0000H	1	C.	BYTE BASED(A)	78	79			
280	0041H	1	C.	BYTE	286	287			
71	0004H	1	C.	BYTE PARAMETER AUTOMATIC	72	73			
			CARRY.	BUILTIN	137	140			
30	0004H	1	CHAR.	BYTE PARAMETER AUTOMATIC	31	32			
38	0034H	15	CHECKCONSOLESTATUS	PROCEDURE BYTE STACK=0008H		324			
107	0016H	1	CHR.	BYTE	109	113	116	119	
216	042FH	59	COMPUTEMONTH. . .	PROCEDURE STACK=0002H	236	SER. =			
202	03FAH	53	COMPUTEYEAR. . .	PROCEDURE STACK=0002H	232				
5			CPMPRODUCT. . . .	LITERALLY	296				
6			CPMVERSION. . . .	LITERALLY	296				
7			CR.	LITERALLY	62	63	284	298	
50	006FH	17	CRLF.	PROCEDURE STACK=000EH		320			
148	0008H	1	D.	BYTE PARAMETER AUTOMATIC	149	151			
277	0000H	2	DATE.	WORD MEMBER(EXTNLTO)		317			
278	0001H	2	DATE.	WORD MEMBER(LCLTOD)	317				
67	0001H	2	DATE.	WORD MEMBER(TOD)	187	230			
225	0024H	1	DATETEST.	BYTE					
176	001BH	1	DAY.	BYTE	183	185	187	237	244
215	0024H	28	DAYLIST.	BYTE ARRAY(28) DATA		241			
2			DCL.	LITERALLY					
118	019BH	17	DEBLANK.	PROCEDURE STACK=0006H		130	150		
232	0586H	65	DISPLAYTOD. . . .	PROCEDURE STACK=0012H		323	330		
83	00F2H	16	EMITBCD.	PROCEDURE STACK=000AH		89	90	99	100
87	0102H	27	EMITBCDPAIR. . . .	PROCEDURE STACK=0010H		94	249		
97	0130H	39	EMITBINPAIR. . . .	PROCEDURE STACK=0010H		104	245		
71	00AEH	28	EMITCHAR.	PROCEDURE STACK=0004H		85	95	105	242
92	011DH	19	EMITCOLON.	PROCEDURE STACK=0016H		247	248		
240	04D4H	77	EMITDATETIME. . .	PROCEDURE STACK=001AH		261			
75	00CAH	40	EMITN.	PROCEDURE STACK=0004H		241			
102	0157H	19	EMITSLANT.	PROCEDURE STACK=0016H		243	244		

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

61	0093H	22	ENTRYERROR	PROCEDURE STACK=000EH	132	135	138	141	145	152	186	271
				311 313								
277	0000H	22	EXTRNLTD.	STRUCTURE BASED(TODPTR)		286	317	319				
24	0000H	1	FCB.	BYTE ARRAY(1) EXTERNAL(4)	303	322	327					
25	0000H	1	FCB16.	BYTE ARRAY(1) EXTERNAL(5)								
4			FOREVER.	LITERALLY 205								
8	0000H	1	FUNC.	BYTE PARAMETER 9								
16	0000H	1	FUNC.	BYTE PARAMETER 17								
20	0000H	1	FUNC.	BYTE PARAMETER 21								
12	0000H	1	FUNC.	BYTE PARAMETER 13								
226	045AH	106	GETDATEIME. . . .	PROCEDURE STACK=0006H	259							
166	0281H	32	GETNEXTDIGIT . . .	PROCEDURE BYTE STACK=0002H								
47	0060H	15	GETSYSDAT.	PROCEDURE POINTER STACK=0008H		301						
108	016AH	49	GNC.	PROCEDURE STACK=0002H	120	142	153					
			HIGH.	BUILTIN 296								
176	001DH	1	HRS.	BYTE 227 247								
67	0003H	1	HRS.	BYTE MEMBER(TOD)	188	227						
277	0002H	1	HRS.	BYTE MEMBER(EXTRNLTD)								
278	0003H	1	HRS.	BYTE MEMBER(LCLTD)								
280	0040H	1	I.	BYTE 285 286 305 306 308								
178	0020H	1	I.	BYTE 181 182 183 193								
55	0014H	1	I.	BYTE 56 57								
70	0015H	1	INDEX.	BYTE 73 79 111 116 260 266 268								
20	0000H	2	INFO.	WORD PARAMETER 22								
16	0000H	2	INFO.	WORD PARAMETER 18								
12	0000H	2	INFO.	WORD PARAMETER 14								
8	0000H	2	INFO.	WORD PARAMETER 10								
293	0042H	1	LASTDSEGBYTE . . .	BYTE INITIAL								
148	0006H	1	LB.	BYTE PARAMETER AUTOMATIC	149	154						
126	0006H	1	LB.	BYTE PARAMETER AUTOMATIC	127	144						
278	0025H	27	LCLTD.	STRUCTURE 283 308 309 310 312 314 317 318 319								
215	0023H	1	LEAPBIAS.	BYTE 220 221 233 235 237								
157	0255H	44	LEAPDAYS.	PROCEDURE BYTE STACK=0006H		187						
178	0021H	1	LEAPFLAG.	BYTE 180 185								
7			LF.	LITERALLY 62 63 298								
3			LIT.	LITERALLY 4 5 6 7 156 279 291								
			LOW.	BUILTIN 296								
167	0019H	1	LSD.	BYTE 168 170								
157	0004H	1	M.	BYTE PARAMETER AUTOMATIC	158	161						
67	0004H	1	MIN.	BYTE MEMBER(TOD)	189	228						
278	0004H	1	MIN.	BYTE MEMBER(LCLTD)								
277	0003H	1	MIN.	BYTE MEMBER(EXTRNLTD)								
176	001EH	1	MIN.	BYTE 228 248								
8	0000H		MON1.	PROCEDURE EXTERNAL(0) STACK=0000H		32	36	45				
12	0000H		MON2.	PROCEDURE BYTE EXTERNAL(1) STACK=0000H			28	39				
16	0000H		MON3.	PROCEDURE WORD EXTERNAL(2) STACK=0000H			42					
20	0000H		MON4.	PROCEDURE POINTER EXTERNAL(3) STACK=0000H				48				
176	001AH	1	MONTH.	BYTE 179 180 182 187 217 218 219 221 237 238 243								
156	0000H	24	MONTHDAYS.	WORD ARRAY(12) DATA	161	187	221	237				
156	0018H	12	MONTHSIZE.	BYTE ARRAY(12) DATA	182							
			MOV8.	BUILTIN 317 319								
			MOVE.	BUILTIN 308								
123	01ACH	17	NUMERIC.	PROCEDURE BYTE STACK=0002H		131	133					
276	0000H	2	OFFSET.	WORD MEMBER(TODPTR)	302							
278	0000H	1	OPCODE.	BYTE MEMBER(LCLTD)	283	309						
67	0000H	1	OPCODE.	BYTE MEMBER(TOD)	190	257	264					
251	0004H	2	PARAMETER.	WORD PARAMETER AUTOMATIC	252	255						

294	05C7H	267	PLM.	PROCEDURE PUBLIC STACK=002EH					
34	0024H	16	PRINTBUFFER. . . .	PROCEDURE STACK=000AH	62	63	298	315	
54	0030H	24	PUTBLANKS.	PROCEDURE STACK=000EH	288				
27	0002H	15	READCONSOLE. . . .	PROCEDURE BYTE STACK=0008H		316	326		
281	0010H	2	RET.	WORD	316	326			
253	000AH	2	RET.	WORD	254	268			
148	0234H	33	SCANDELIMITER. . .	PROCEDURE BYTE STACK=0020H		183	184	189	193 196
126	01BDH	119	SCANNUMERIC. . . .	PROCEDURE BYTE STACK=0016H		154	179	188	
67	0005H	1	SEC.	BYTE MEMBER(TOD)	194	196	229		
278	0005H	1	SEC.	BYTE MEMBER(LCLTOD)					
277	0004H	1	SEC.	BYTE MEMBER(EXTRNLTOD)					
176	001FH	1	SEC.	BYTE	229	249			
276	0002H	2	SEGMENT.	WORD MEMBER(TODPTR)					
177	02C8H	289	SETDATETIME. . . .	PROCEDURE STACK=0024H		267			
			SHL.	BUILTIN	136	174	200	241	
			SHR.	BUILTIN	89	160			
67	0000H	1	STRING.	BYTE BASED(STRINGADR) ARRAY(1)		73	79	116	266 268
68	0002H	2	STRINGADR.	WORD	69	73	79	116	256 266 268
44	0052H	14	TERMINATE.	PROCEDURE STACK=0008H		64	299	331	
225	0008H	2	TESTVALUE.	WORD					
1	0002H		TOD.	PROCEDURE STACK=0000H					
67	0000H	27	TOD.	STRUCTURE BASED(TODADR)		187	188	189	190 194 196 227 228
					229	230	256	257	264
66	0000H	2	TODADR.	WORD	67	187	188	189	190 194 196 227 228 229 230 255
					256	257	264		
251	0521H	101	TODASCII.	PROCEDURE STACK=002AH		314			
279			TODELIMITER. . . .	LITERALLY	310				
275	000CH	4	TODPOINTER.	POINTER	276	277	286	301	317 319
276	000CH	4	TODPTR.	STRUCTURE AT	302				
291			TODSDOFFSET. . . .	LITERALLY	302				
148	0004H	1	UB.	BYTE PARAMETER AUTOMATIC		149	154		
126	0004H	1	UB.	BYTE PARAMETER AUTOMATIC		127	144		
172	0004H	1	VAL.	BYTE PARAMETER AUTOMATIC		173	174		
292	0012H	2	VERS.	WORD	295	296			
41	0043H	15	VERSION.	PROCEDURE WORD STACK=0008H		295			
215	0022H	1	WEEKDAY.	BYTE	231	241			
165	0004H	2	WORDVALUE.	WORD	168	169	209	211	221 230 231 234 237
30	0011H	19	WRITECONSOLE. . . .	PROCEDURE STACK=000AH		51	52	58	284 287
157	0006H	1	Y.	BYTE PARAMETER AUTOMATIC		158	160	161	
176	001CH	1	YEAR.	BYTE	184	185	187	204	207 212 234 245
203	0006H	2	YEARLENGTH.	WORD	206	208	209	211	
159	0018H	1	YP.	BYTE	160	162	163		

MODULE INFORMATION:

CODE AREA SIZE = 06D2H 1746D
 CONSTANT AREA SIZE = 00BEH 190D
 VARIABLE AREA SIZE = 0043H 67D
 MAXIMUM STACK SIZE = 002EH 46D
 513 LINES READ
 0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #